

Paper AD03

%XMLPORT: An XML Substitute for SAS® PROC CPORT/CIMPORT (almost)

Andy Barnett, CSG, Inc., Raleigh, NC

ABSTRACT

With the introduction of XML and other specifications, previous means of transporting SAS® libraries in “generic” form are being replaced. This paper presents a macro, %XMLPORT, intended to provide an XML solution to the SAS PROCs, CPORT and CIMPORT. At a minimum, the specified library's data sets (both metadata and data) and format catalog entries are converted to a single XML file, which can then be re-assembled on a different host. The process is tested as follows: use %XMLPORT on a test library, transmit the XML file to the new host; use %XMLPORT (in “reverse”) to rebuild the SAS library there; use %XMLPORT to build a 2nd XML file; transmit this new file back to the original host; use %XMLPORT again to rebuild a copy of the original library on the original host; and finally use a comparison procedure such as PROC COMPARE to show the two libraries are equivalent.

INTRODUCTION

This seems to be a rather difficult problem. As my first exploration into XML, I explored a number of avenues. Each had drawbacks that prevented the method from working easily. The desired action is to produce a single XML file from the original SAS library of data sets and format catalog(s), and then to reproduce the original files as defined by a PROC COMPARE using the EXACT method. The approach finally chosen for this paper is to convert format catalogs to CNTLOUT data sets, then combine all the data sets into one special normalized data set, and finally export that via LIBNAME libref XML and PROC COPY. Then, to import the data, the single XML file is copied back into a SAS data set by the same means, the original tables and catalogs reconstructed, and then some cleanup to eliminate the unwanted overhead created along the way.

The main problem encountered was that the XML engine, without a special user-created TAGSET or XMLMAP, could only persist certain parts of the metadata and even data. On comparison, some variables changed TYPE from character to numeric, or received new formats, informat, and default labels (the same as the variable name).

Thus, one solution was to use SAS' own metadata DICTIONARY tables and include that in the normalized master data set. However, this puts more metadata inside of XML metadata, and is not the elegant approach I would have preferred.

%NORMALLIB

This submacro first creates a list (METADATA) of all the applicable metadata needed to recreate the tables and format catalogs in the specified library (&LIB). But since some of this metadata is LABEL and MEMLABEL data that may have characters such as “&” that XML requires to be quoted, and since the table and format catalog character data themselves may have such special characters, I decided to add the METADATA table and the &LIB tables and catalogs to a copy (TEMPLIB1) of &LIB and work from there.

Now, we produce a new METADATA table from TEMPLIB1, but this will not be transmitted via the XML file. Instead, it is used to construct a %INCLUDE file to “quote” the character data and to normalize all the tables and catalogs into a single huge data set, from which we can easily write an XML file. (This XML file will contain the original METADATA file on &LIB, not on TEMPLIB1, which will be used on the far side to reconstruct certain attributes, and then be discarded.)

The “quoting” consists of taking the numeric data and converting each max-8-byte field to a HEX16. character VALUE. Similarly, each character variable is converted 8-bytes-at-a-time into a corresponding \$HEX16. character VALUE. So the two types of data are saved in the same column. There is a LIBNAME MEMNAME MEMTYPE RCDSEQ VARNUM NAME TYPE LENGTH FORMAT INFORMAT COLSEQ set of key variables attached to each VALUE, and this is what the final normalized, quoted data set looks like (which is called MASTER).

This is now copied via the LIBNAME XML-engine into the specified XML file, containing the entire &LIB tables and format catalogs, as well as a copy of the &LIB METADATA table to help out on the far side.

%DENORMALLIB

Then, the reverse step on the far side can be handled similarly by first importing the XML file into a new copy of the master table. This table may lose some of its original attributes (which is what spawned this method in the first

place!), but the MASTER table contains all the information needed to fix these problems in the contained tables (namely in the contained METADATA table), and then we no longer need the MASTER or METADATA tables.

So, we import the XML to the MASTER table, set a macro variable BY=LIBNAME MEMNAME MEMTYPE RCDSEQ VARNUM NAME TYPE LENGTH FORMAT INFORMAT, sort the master table BY &BY COLSEQ, and then do the equivalent of a transpose BY &BY, using VAR VALUE (which at this point is all character), and ID COLSEQ.

Then the submacro rebuilds the original tables, using the metadata to guide it.

Any format catalogs are rebuilt by converting the CNTLOUT data sets into format catalogs, and discarding the CNTLOUT data sets. (This is of course how many users solved the data-set-only problem of PROC COPY to transfer format catalogs). Then, some housecleaning is performed, and the final library has been rebuilt.

DOES THIS WORK?

Now we need to test this, so we create a few data sets and two format catalogs and store them in library WORK1. Then we use PROC CPORT to copy this to a CPT file, and import that back into WORK2 via PROC CIMPORT. Then, we use %XMLPORT to create an XML file, and then to import that back into WORK3. And, finally, we use a PROC COMPARE tool to compare libraries (checks all metadata and data, and reports on one page the overall results as counts of errors found).

As can be seen in Appendices A and B, the comparisons of WORK1 with WORK2 and WORK1 with WORK3 both work as expected.

What is perhaps most interesting is that %XMLPORT may do better than PROC CPORT/CIMPORT for certain situations, at least with the options I used. The LARGE and SMALL numbers were not transported precisely by CPORT, though they are still essentially very large and very small; also the special missing values were not preserved.

However, compared to CPORT, %XMLPORT has an anomaly in the LENGTH attribute of the format catalogs, where %XMLPORT increases this by 1, whereas PROC CPORT/CIMPORT does not.

Whether %XMLPORT will succeed on other non-PC platforms is not yet known, since they may use internal numeric representations different from the IEEE standard used on PCs, or may not use the ASCII character set. But I suspect a suitable format/informat in the decoding part of %XMLPORT (i.e., DENORMALLIB) could fix this.

A better solution is probably found in using a special tagset, which LIBNAME XML allows via the TAGSET= option.

CONCLUSION

Using XML can be a daunting task. I'm sure that is a measure of the power of the language, and I would have preferred to produce a more completely XML-dependent solution using tagsets, which SAS provides the capability to do. Whether or not this XML-solution is useful remains to be seen.

REFERENCES

- SAS Institute Inc. 2007. **SAS OnlineDoc® 9.1.3**. Cary, NC: SAS Institute Inc. (see <http://support.sas.com/onlinedoc/913/docMainpage.jsp>)

ACKNOWLEDGMENTS

I thank SAS Institute for help delving into experimental tagsets, and numerous folks on SAS-L who have described a number of solutions to at least parts of this particular problem.

I also thank Kim Sturgen (PPD, Inc.) and Christine Ward (CSG, Inc.), as well as the SESUG 2007 staff, particularly Denise Kruse, Ron Fehd, and Ed Heaton.

This macro was coded and tested using SAS/Learning Edition 4.1, which though limited to 1500 output observations per step, is still quite powerful, and "includes Base SAS 9.1.3, SAS/STAT, SAS/GRAPH, SAS/QC, SAS/ETS and SAS Enterprise Guide 4.1" and *The Little SAS Book for Enterprise Guide 4.1* (see <http://support.sas.com/le>) . It runs on Windows 2000 and Windows XP (including XP Pro x64). I highly recommend SAS/LE to anyone needing a non-commercial license to learn more about these components of SAS.

RECOMMENDED READING

- Richard A. DeVenezia's macros on his website, particularly %XMLIB (see <http://www.devenezia.com/downloads/sas/macros/index.php?m=xmlib>)
- <http://support.sas.com/rnd/base/index-xml-resources.html>
- <http://support.sas.com/rnd/base/topics/odsmarkup/tagsets.html>
- David Goldberg, *What Every Computer Scientist Should Know About Floating-Point Arithmetic*, ACM Computing Surveys, Vol. 23, No. 1, March 1991, pp. 5-48.

CONTACT INFORMATION

Andy Barnett, Sr. Programmer Analyst
CSG, Inc.
c/o Christine Ward, Sr. Recruiter
4300 Six Forks Road
Suite 720
Raleigh, NC 27609-5764
919-839-8113
Fax: 919-839-8114
Web: <http://www.csolutionsgroup.com>
Andy's Home Email: aebarnet@nc.rr.com, phone: 919-850-2196

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

APPENDIX A - %LIBCOMP RESULTS OF COMPARING WORK1 AND WORK2

```
LIBCOMP2: IGNORECASE=N, IGNOREVARS= (METADATA)
BASE=LIBDEV(C:\DOCUME~1\barretta\LOCALS~1\Temp\SAS Temporary Files\_TD3580\work1),
COMPARE=LIBVAL(C:\DOCUME~1\barretta\LOCALS~1\Temp\SAS Temporary Files\_TD3580\work2)
PRINT: DSM=WORK.CLS_DIFF(''), NOBS=6, PROBS=100
CLS DIFF: Data diffs for BASE=WORK1.CLASS1 vs. COMPARE=WORK2.CLASS1
```

Obs	_type_	COMMENT
1	C	I=1: THIS IS A FAIRLY LONG COMMENT: ALL RULES HAVE EXCEPTIONS EXCEPT FOR THIS ONE.
2	B	I=1: THIS IS A FAIRLY LONG COMMENT: ALL RULES HAVE EXCEPTIONS EXCEPT FOR THIS ONE.
3	C	I=28: THIS IS A FAIRLY LONG COMMENT: ALL RULES HAVE EXCEPTIONS EXCEPT FOR THIS ONE.
4	B	I=28: THIS IS A FAIRLY LONG COMMENT: ALL RULES HAVE EXCEPTIONS EXCEPT FOR THIS ONE.
5	C	I=2: THIS IS A FAIRLY LONG COMMENT: ALL RULES HAVE EXCEPTIONS EXCEPT FOR THIS ONE.
6	B	I=2: THIS IS A FAIRLY LONG COMMENT: ALL RULES HAVE EXCEPTIONS EXCEPT FOR THIS ONE.

Obs	DATE	DATETIME	I	IS8601	LARGE	LEADINGSPACE
1	17370	1500849394.4	1	1500849394.4	7.237E75	1
2	17370	1500849394.4	1	1500849394.4	1E301	1
3	17370	1500849394.4	28	1500849394.4	7.237E75	28
4	17370	1500849394.4	28	1500849394.4	1E301	28
5	17370	1500849394.4	2	1500849394.4	7.237E75	2
6	17370	1500849394.4	2	1500849394.4	1E301	2

[illegible]

Obs	MISSING	NAME	SMALL	TIME	lc_seq	_cdummy	_obs_
1		THIS IS THE NAME VAR	-5.3976E-79	81394.38	1		1
2	.	THIS IS THE NAME VAR	-1E-299	81394.38	1		2
3	Z	THIS IS THE NAME VAR	-5.3976E-79	81394.38	1		3
4	.	THIS IS THE NAME VAR	-1E-299	81394.38	1		4

```

5      .      THIS IS THE NAME VAR      -5.3976E-79      81394.38      1      5
6      .      THIS IS THE NAME VAR      -1E-299      81394.38      1      6

```

APPENDIX B - %LIBCOMP RESULTS OF COMPARING WORK1 AND WORK3

```

LIBCOMP2: IGNORECASE=N, IGNOREVARS= (METADATA)
BASE=LIBDEV(C:\DOCUME~1\barnetta\LOCALS~1\Temp\SAS Temporary Files\_TD3580\work1),
COMPARE=LIBVAL(C:\DOCUME~1\barnetta\LOCALS~1\Temp\SAS Temporary Files\_TD3580\work3)
PRINT: DSN=WORK.LC_SUMMARY( ), NOBS=4, PROBS=MAX

```

Obs	memname	base	compare	structdiffs	datadiffs	nvarbase	nvarcomp	nobsbase	nobscomp	totaldiffs	nerotypes
1	CLASS1	WORK1	WORK3	.	0	12	12	3	3	0	0
2	NOOBS	WORK1	WORK3	.	0	1	1	0	0	0	0
3	NOTHING	WORK1	WORK3	.	.	0	0	0	0	0	0
4	NOVARS	WORK1	WORK3	.	.	0	0	1	1	0	0
				=====	=====	=====	=====	=====	=====	=====	=====
				0	0	13	13	4	4	0	0

APPENDIX C - %XMLPORT MACRO (UNFINISHED — WILL BE REFORMATTED AND FINISHED SOON)

```

options ls = 132 ps = 60 compress = yes spool noxwait xsync errorabend noovp nomlogic nosymbolgen
mprint mprintnest nocenter;
%let work1 = %sysfunc(pathname(work))\work1;
%let work2 = %sysfunc(pathname(work))\work2;
%let work3 = %sysfunc(pathname(work))\work3;
%let work = %sysfunc(pathname(work));

x "mkdir "&work1"";
x "mkdir "&work2"";
x "mkdir "&work3"";

libname work1 "&work1";
libname work2 "&work2";
libname work3 "&work3";
filename cpt "&work\work1.cpt";
libname xml xml "&work\work1.xml";

%let cpt = %sysfunc(pathname(cpt));
%let xml = %sysfunc(pathname(xml));

filename tmp "&work\tmp.tmp";

%let border = %sysfunc(repeat(-,132-1));
%let lsfmt = $132.;
title "XMLPORT";
title2 "&border";
%let ntitle = 2;
footnote "&border";
footnote2 %unquote(%qsysfunc(quote(%qsysfunc(dequote(%superq(sysprocessname))))));
footnote3 "%sysfunc(putc(Rundate: &sysdate9:&sysptime - SAS &sysvlong on &syssscl1,&lsfmt))";
footnote4 "%sysfunc(putc(WORK1=&work1,&lsfmt))";
footnote5 "%sysfunc(putc(WORK2=&work2,&lsfmt))";
footnote6 "%sysfunc(putc(WORK3=&work3,&lsfmt))";
footnote7 "%sysfunc(putc(CPT=&cpt,&lsfmt))";
footnote8 "%sysfunc(putc(XML=&xml,&lsfmt))";
%let nfoot = 8;

*These macros not shown due to space - most may be found in PharmaSUG 2006 Paper AD04 by Andy Barnett;
%include 'macros_sesug2007.sas' / nosource2; *These macros excluded for brevity;

%*-----;
%* MACRO XMLPORT
%*-----;
%macro xmlport( lib, selmems =, excmems =, prefix = xtmp, master = &prefix._master, xmlref = xml, inout = OUT,
metadata = &prefix._metadata );

%*=====;
%* See START HERE below...
%*=====;

%local id err dbg kerr;
%let id = %sysmacroname;
%let err = E%str(RROR): &id;
%let dbg = DEBUG: &id;
%let kerr = 0;

%let lib = %qupcase(&lib);
%let prefix = %qupcase(&prefix);
%let prefix_escaped = %qsysfunc(tranwrd(&prefix,_,^_));
%let uqp = %unquote(&prefix);
%let inout = %qupcase(&inout);

%global &uqp._maxcharlen &uqp._maxcharlen8;
%let &uqp._maxcharlen = 32767;
%let &uqp._maxcharlen8 = %sysfunc(int(%sysevalf((&&uqp._maxcharlen+7)/8)));

```

```

%PUT &UQP._MAXCHARLEN=&&&UQP._MAXCHARLEN, &UQP._MAXCHARLEN8=&&&UQP._MAXCHARLEN8;

%*-----;
%* SUBMACRO NORMAL: Convert all data sets in library &LIB to special normalized form in WORK area
%*-----;
%macro normal( lib, selmems =, excmems =, prefix = xttmp, master = &prefix._master, xmlref = xml,
    metadata = &prefix._metadata, templib1 = &prefix._l1 );

    %local id err dbg kerr;
    %let id = &sysmacroname;
    %let err = E%str(RROR): &id;
    %let dbg = DEBBUG: &id;
    %let kerr = 0;

    %let uqp = %unquote(&prefix);
    %global &uqp._maxcharlen &uqp._maxcharlen8;

%macro meta( lib, pfx = XTMP, out = &pfx._metadata );
    %let lib = %upcase(&lib);

%*Get list of table and column metadata in library &lib;
proc sql;
    create table work.&pfx._normal_tables as
    select *
    from dictionary.tables
    where libname = "%unquote(&lib)"
    order by memname;

    create table work.&pfx._normal_columns as
    select *
    from dictionary.columns
    where libname = "%unquote(&lib)"
    order by memname, varnum;

    create table work.&pfx._normal_catalogs as
    select distinct libname, memname, memtype
    from dictionary.catalogs
    where libname = "%unquote(&lib)" and objtype in ( "FORMAT" "FORMATC" "INFMT" "INFMTC" )
    order by memname;

%put OUT=&out;
%*Combine lists into one;

    create table work.&out ( drop = _ : ) as
    select coalesce( a._libname1, b._libname ) as libname,
        coalesce( a._memname1, b._memname ) as memname,
        coalesce( a._memtype1, b._memtype ) as memtype,
        b.*
    from work.&pfx._normal_catalogs
        ( rename = ( libname = _libname1 memname = _memname1 memtype = _memtype1 ) ) a
    full join
        ( select coalesce( a._libname1, b._libname2 ) as _libname length = 8,
            coalesce( a._memname1, b._memname2 ) as _memname length = 32,
            coalesce( a._memtype1, b._memtype2 ) as _memtype length = 12,
            a.*, b.*
        from work.&pfx._normal_tables
            ( rename = ( libname = _libname1 memname = _memname1 memtype = _memtype1 ) ) a
        full join
            work.&pfx._normal_columns
            ( rename = ( libname = _libname2 memname = _memname2 memtype = _memtype2 ) ) b
        on a._libname1 = b._libname2 and a._memname1 = b._memname2 and a._memtype1 = b._memtype2 ) b
    on a._libname1 = b._libname and a._memname1 = b._memname and a._memtype1 = b._memtype
    order by libname, memname, memtype, memlabel, varnum, name, type, length, format, informat, label;
quit;
%mend meta;

*METADATA is the list of metadata, whereas MASTER is the normalized quoted data - some of the metadata needs to
be quoted, so I need to tack it on to the end of the MASTER table, and then dequote it on import, and use that
to assign certain quoted attributes to the final denormalized data - so, lets add this table to the temporary
output copy, and then sneak it through and remove it on the other side;

%let metadata = %unquote(&metadata);
%meta( &lib, out = &metadata )

%let templib1 = %unquote(&templib1);
proc copy in = &lib out = &templib1 mt = data; run;

*Copy format catalogs to &TEMPLIB1 as CNTLOUT data sets;
%local cats kcats;
%let cats =;
%let kcats = 0;
proc sql noprint;
    select strip( memname ) as memname, count(*) as kcats
    into :cats separated by ' ', :kcats
    from ( select distinct memname
        from dictionary.catalogs

```

```

        where libname = "&lib" and memtype = 'CATALOG'
            and upcase( objtype ) in ( "FORMAT" "FORMATC" "INFMT" "INFMTC" ) )
order    by memname;
quit;

%local cat i;
%put KCATS=&kcats, CATS=&cats;
%do i = 1 %to &kcats;
    %let cat = %scan(&cats,&i,%str( ));
    %put I=&i, CAT=&cat, TEMPLIB1.CAT=&templib1.&cat;
    proc format maxlablen = 200 library = work1.&cat cntlout = &templib1.&cat; run;
    proc sort data = &templib1.&cat; by fmtname type hlo start end; run;
%end;

proc copy in = work out = &templib1; select &metadata; run;

%meta( &templib1, out = &metadata )

%*Create code to normalize each table;
%if %sysfunc(exist(work.%unquote(&master))) = 1 %then %do;
    proc delete data = work.%unquote(&master); run;
%end;

data _null_;
    file &uqp._1;
    set work.&metadata;
    by libname memname memtype varnum;
    if first.memname then do;
        array cmetas $&&uqp._maxcharlen nvars nvarnums nlengths nformats ninformats cvars cvarnums clengths
                                cformats cinformats;

        do over cmetas; cmetas = ' '; end;
        retain cmetas;
        knvars = 1;
        kcvvars = 1;
    end;
    if format = ' ' then format = "@";
    if informat = ' ' then informat = "@";
    if type = 'num' then do;
        nvars = left( strip( nvars ) || ' ' || name );
        nvarnums = left( strip( nvarnums ) || ' ' || compress( put( varnum, best. ) ) );
        nlengths = left( trim( nlengths ) || ' ' || compress( put( length, best. ) ) );
        nformats = left( trim( nformats ) || ' ' || format );
        ninformats = left( trim( ninformats ) || ' ' || informat );
        knvars + 1;
    end;
    else if type = 'char' then do;
        cvars = left( strip( cvars ) || ' ' || name );
        cvarnums = left( strip( cvarnums ) || ' ' || compress( put( varnum, best. ) ) );
        clengths = left( trim( clengths ) || ' ' || compress( put( length, best. ) ) );
        cformats = left( trim( cformats ) || ' ' || format );
        cinformats = left( trim( cinformats ) || ' ' || informat );
        kcvvars + 1;
    end;
    else if type ne ' ' or nvar ne 0 then abort abend;
    if last.memname then do;
        put %prtemp( ' libname +(-1) ' memname
                    ' , titles = "NORMAL: Dump of ' libname +(-1) ' memname " )'
            /"data work.&uqp._tmp ( keep = %prefix(libname memname memtype varnum name type length varlen "
            "format informat rcdseq colseq value,&uqp._) );"
            /" length"
            /" &uqp._libname $8 &uqp._memname $32 &uqp._memtype $12 &uqp._name $32"
            /" &uqp._type $4 &uqp._length &uqp._varlen &uqp._varnum 8"
            /" &uqp._format &uqp._informat $32"
            /" &uqp._rcdseq &uqp._colseq 8 &uqp._value $16"
            /" &uqp._cdummy $1 &uqp._ndummy 8 &uqp._tmp $&&uqp._maxcharlen;"
            /" retain &uqp._libname " libname " &uqp._memname " memname " "
            "&uqp._memtype " memtype " &uqp._cdummy " &uqp._ndummy .;"
            /" set " libname +(-1) ' memname ";"
            /" array &uqp._nvars 8 ";
        do i = 1 to length( nvars ) while ( scan( nvars, i, ' ' ) ne ' ' );
            tmp = scan( nvars, i, ' ' );
            put ' ' tmp;
        end;
        put " &uqp._ndummy;"
            /" array &uqp._nnames(" knvars ' ) $32 _temporary_ ( '
            ;
        do i = 1 to length( nvars ) while ( scan( nvars, i, ' ' ) ne ' ' );
            tmp = quote( scan( nvars, i, ' ' ) );
            put ' ' tmp;
        end;
        put ' " " );'
            /" array &uqp._nvarnums(" knvars " ) 8 _temporary_ ( ";
        do i = 1 to length( nvarnums ) while ( scan( nvarnums, i, ' ' ) ne ' ' );
            tmp = scan( nvarnums, i, ' ' );
            put ' ' tmp;
        end;
    end;
end;

```

```

put ' . );'
/" array &uqp._nlengths(" knvars ") 8 _temporary_ ( ";
do i = 1 to length( _nlengths ) while ( scan( _nlengths, i, ' ' ) ne ' ' );
tmp = scan( _nlengths, i, ' ' );
put ' ' tmp;
end;
put ' . );'
/" array &uqp._nformats(" knvars ") $32 _temporary_ ( ";
do i = 1 to length( _nformats ) while ( scan( _nformats, i, ' ' ) ne ' ' );
tmp = quote( scan( _nformats, i, ' ' ) );
put ' ' tmp;
end;
put ' " " );'
/" array &uqp._ninformats(" knvars ") $32 _temporary_ ( ";
do i = 1 to length( _ninformats ) while ( scan( _ninformats, i, ' ' ) ne ' ' );
tmp = quote( scan( _ninformats, i, ' ' ) );
put ' ' tmp;
end;
put ' " " );'
/" array &uqp._cvars 8 ";
do i = 1 to length( _cvars ) while ( scan( _cvars, i, ' ' ) ne ' ' );
tmp = scan( _cvars, i, ' ' );
put ' ' tmp;
end;
put " &uqp._cdummy;"
/" array &uqp._cnames(" kcvars ") $32 _temporary_ ( ";
do i = 1 to length( _cvars ) while ( scan( _cvars, i, ' ' ) ne ' ' );
tmp = quote( scan( _cvars, i, ' ' ) );
put ' ' tmp;
end;
put ' " " );'
/" array &uqp._cvarnums(" kcvars ") 8 _temporary_ ( ";
do i = 1 to length( _cvarnums ) while ( scan( _cvarnums, i, ' ' ) ne ' ' );
tmp = scan( _cvarnums, i, ' ' );
put ' ' tmp;
end;
put ' . );'
/" array &uqp._clengths(" kcvars ") 8 _temporary_ ( ";
do i = 1 to length( _clengths ) while ( scan( _clengths, i, ' ' ) ne ' ' );
tmp = scan( _clengths, i, ' ' );
put ' ' tmp;
end;
put ' . );'
/" array &uqp._cformats(" kcvars ") $32 _temporary_ ( ";
do i = 1 to length( _cformats ) while ( scan( _cformats, i, ' ' ) ne ' ' );
tmp = quote( scan( _cformats, i, ' ' ) );
put ' ' tmp;
end;
put ' "@ " );'
/" array &uqp._cinformats(" kcvars ") $32 _temporary_ ( ";
do i = 1 to length( _cinformats ) while ( scan( _cinformats, i, ' ' ) ne ' ' );
tmp = quote( scan( _cinformats, i, ' ' ) );
put ' ' tmp;
end;
put ' "@ " );'
/" &uqp._rcdseq + 1;"
/" do &uqp._i = 1 to dim( &uqp._nvars ) - 1;"
/" &uqp._name = &uqp._nnames(&uqp._i);"
/" &uqp._type = 'N';"
/" &uqp._varnum = &uqp._nvarnums(&uqp._i);"
/" &uqp._length = &uqp._nlengths(&uqp._i);"
/" &uqp._format = &uqp._nformats(&uqp._i);"
/" &uqp._informat = &uqp._ninformats(&uqp._i);"
/" &uqp._colseq = 1;"
/" if nmiss( &uqp._nvars(&uqp._i) ) = 0"
/" then &uqp._value = strip( put( &uqp._nvars(&uqp._i), ??hex16. ) );"
/" else do;"
/" if &uqp._nvars(&uqp._i) = . then &uqp._value = '.';"
/" else &uqp._value = '.' || strip( put( &uqp._nvars(&uqp._i), ??hex16. ) );"
/" end;"
/" &uqp._varlen = length( &uqp._value );"
/" output;"
/" end;"
/" do &uqp._i = 1 to dim( &uqp._cvars ) - 1;"
/" &uqp._name = &uqp._cnames(&uqp._i);"
/" &uqp._type = 'C';"
/" &uqp._varnum = &uqp._cvarnums(&uqp._i);"
/" &uqp._length = &uqp._clengths(&uqp._i);"
/" &uqp._format = &uqp._cformats(&uqp._i);"
/" &uqp._informat = &uqp._cinformats(&uqp._i);"
/" &uqp._tmp = &uqp._cvars(&uqp._i);"
/" &uqp._varlen = length( &uqp._tmp );"
/" &uqp._tmplen = &uqp._varlen;"
/" &uqp._colseq = 0;"
/" do &uqp._j = 1 to &uqp._varlen while ( &uqp._tmplen > 0 );"
/" &uqp._minlen = min( &uqp._tmplen, 8 );"

```

```

        /"      &uqp._value = substr( &uqp._tmp, 1, &uqp._minlen );"
        /"      length &uqp._fmt $32;"
        /"      &uqp._fmt = '$hex' || strip( put( &uqp._minlen * 2, best. ) ) || '.';"
        /"      &uqp._value = putc( substr( &uqp._value, 1, &uqp._minlen ), &uqp._fmt );"
        /"      &uqp._colseq + 1;"
        /"      output;"
        /"      &uqp._tmplen = &uqp._tmplen - &uqp._minlen;"
        /"      &uqp._tmp = substrn( &uqp._tmp, &uqp._minlen + 1, &uqp._tmplen );"
        /"      end;"
        /"      end;"
        /"run;"
        /"proc append data = work.&uqp._tmp out = work.%unquote(&master); run;"
        ;
    end;
run;

%filedump( &uqp._1 )

%include &uqp._1;

proc sort data = work.%unquote(&master);
    by %prefix( libname memname memtype rcdseq varnum name type length varlen format informat colseq,&uqp._ );
run;

%prtempty( work.%unquote(&master), titles = "NORMAL: End" )

*We now have the metadata and the data all in one table;
%put &id: PROC COPY: SELECT &master;
proc copy in = work out = &xmlref;
    select %unquote(&master);
run;

%goto fin;

%err:
    %put &err: Aborting...;
    %put &err: _local_;
    %unquote(e%str(ndsas;))

%fin:
    %put &id: End;
%mend normal;

%*-----;
%* SUBMACRO DENORMAL: Convert special normalized form back to all original data sets in new lib &LIB
%*-----;
%macro denormal( lib, selmems =, excmems =, prefix = xtmp, xmlref = xml, master = %prefix._master,
    metadata = %prefix._metadata, templibl = %prefix._l1 );

    %local id err dbg kerr;
    %let id = &sysmacroname;
    %let err = E%str(RROR): &id;
    %let dbg = DEBUG: &id;
    %let kerr = 0;

    %let uqp = %unquote(&prefix);
    %global &uqp._maxcharlen &uqp._maxcharlen8;

    %let metadata = %unquote(&metadata);

%*Check that no members exist in &LIB library - abort if they do;
    %prtempty( sashelp.vtable, where = libname = "%unquote(&lib)", titles = "DENORMAL at startup" )
    %local count;
    %let count = 0;
    proc sql noprint;
        select count(*) as count into :count from dictionary.tables where libname = "%unquote(&lib)";
    quit;
    %put COUNT=&count;
    %if &count ne 0 %then %goto err;

%*Extract the XML file into a SAS table named &master;
    %put &id: PROC COPY: SELECT &master;
    proc copy in = &xmlref out = work;
        select &master;
    run;

%*Generate code to recreate tables in &LIB library from MASTER;
    proc sort data = work.&master;
        by %prefix(libname memname memtype rcdseq varnum name type length varlen format informat colseq,&uqp._ );
    run;

    %prtempty( work.&master, titles = "DENORMAL: WORK.&MASTER" )

%*Rebuild the original MASTER data table;
    data work.&uqp._tmp ( keep = %prefix(libname memname memtype rcdseq varnum name type length varlen

```

```

        format informat,&uqp._) val_1 - val_&&uqp._maxcharlen8 );
set work.&master;
by %prefix(libname memname memtype rcdseq varnum name type length varlen format informat colseq,&uqp._);
array vals $16 val_1 - val_&&uqp._maxcharlen8;
retain vals;
if first.&uqp._informat then do i = 1 to &&uqp._maxcharlen8; vals(i) = ' '; end;
vals(&uqp._colseq) = &uqp._value;
if last.&uqp._informat;
run;

/*
%prtempty( work.&uqp._tmp, titles = "DENORMAL: Transpose of WORK.&MASTER" )
*/

%*Now, if TYPE=C, then combine VAL_1 - up to VAL_&&uqp._maxcharlen8 hex-encoded $16s into one long
$&&uqp._maxcharlen VALUE (padded with blanks), and will have LENGTH=LENGTH, and actual length of VARLEN (may
have leading significant blanks). If TYPE=N, then only VAL_1 should be populated, and we convert it to numeric
with INPUT using HEX16.;
%*NOTE: On some platforms, it may not work to use HEX16 - have only tried this on PCs, where it works with
every number I have tried - but of course, that is not a proof that it will always work.;

data &uqp._tmp ( keep = %prefix(libname memname memtype rcdseq varnum name type length varlen format
informat nvalue value,&uqp._) );
set &uqp._tmp ( keep = %prefix(libname memname memtype rcdseq varnum name type length varlen format
informat,&uqp._) val_1 - val_&&uqp._maxcharlen8 );
by %prefix(libname memname memtype rcdseq varnum name type length varlen format informat,&uqp._);
array vals $16 val_1 - val_&&uqp._maxcharlen8;
length &uqp._nvalue 8 &uqp._value $&&uqp._maxcharlen fmt $32;
retain &uqp._value vals ' ';
&uqp._value = ' ';
&uqp._nvalue = .;
if &uqp._type = 'C' then do i = 0 to %eval(&&uqp._maxcharlen8-1) while ( vals(i+1) ne ' ' );
nchars = min( 8, &uqp._varlen - i*8 );
fmt = '$hex' || strip( put( 2 * nchars, best. ) ) || '.';
&uqp._value = substrn( &uqp._value, 1, i*8 ) || inputc( vals(i+1), fmt );
put &uqp._memname= &uqp._rcdseq= &uqp._varnum= &uqp._name= &uqp._type= fmt= vals(i+1)= &uqp._value=;
end;
else do;
&uqp._nvalue = input( val_1, ??hex16. );
put &uqp._memname= &uqp._rcdseq= &uqp._varnum= &uqp._name= &uqp._type= val_1= &uqp._nvalue=;
end;
run;

%prtempty( &uqp._tmp,
titles = "DENORMAL: AFTER TRANSPOSE AND CONCATENATE",
rptstmts = %quote(
by &uqp._libname &uqp._memname &uqp._memtype;
column &uqp._rcdseq &uqp._varnum &uqp._name &uqp._type &uqp._length &uqp._varlen
&uqp._format &uqp._informat &uqp._nvalue &uqp._value;
define &uqp._rcdseq / order width = 6 "RCDSEQ";
define &uqp._varnum / order width = 6 "VARNUM";
define &uqp._name / order width = 12 flow "NAME";
define &uqp._type / order width = 1 "TYPE";
define &uqp._length / order width = 6 "LENGTH";
define &uqp._varlen / order width = 6 "VARLEN";
define &uqp._format / order width = 12 flow "FORMAT";
define &uqp._informat / order width = 12 flow "INFORMAT";
define &uqp._nvalue / display width = 12 format = best12. "NVALUE";
define &uqp._value / display width = 30 flow "VALUE";
) )

%*Now we can create the final data sets and format catalogs as defined in this MASTER table;
proc sql noprint;
create table work.&uqp._tmp2 as
select distinct %comma( %prefix( libname memname memtype varnum name type length format informat,&uqp._) )
from work.&uqp._tmp
order by %comma( %prefix( libname memname memtype varnum name type length format informat,&uqp._) );

create index libmem on work.&uqp._tmp2 ( &uqp._libname, &uqp._memname );

select &uqp._libname, upcase( &uqp._memname ) as memname, count(*) as kmems
into :libnames separated by ' ', :memnames separated by ' ', :kmems
from ( select distinct * from work.&uqp._tmp2 ( keep = &uqp._libname &uqp._memname ) )
order by &uqp._libname, &uqp._memname;
quit;

%put KMEMS=&kmems, MEMNAMES=&memnames;

%do i = 1 %to &kmems;
%let libname = %scan(&libnames,&i,%str( ));
%let memname = %scan(&memnames,&i,%str( ));

data _null_;
length %prefix(keep= num= char= tmp, &uqp._) $&&uqp._maxcharlen;
set &uqp._tmp2;
by %prefix(libname memname memtype varnum name type length format informat,&uqp._);

```

```

where &uqp._libname = "&libname" and upcase( &uqp._memname ) = "&memname";
retain %prefix(keeps nums chars, &uqp._ ) ' ';
if first.&uqp._memname then do;
    &uqp._keeps = ' ';
    &uqp._nums = ' ';
    &uqp._chars = ' ';
    file &uqp._1;
*NOTE: We write the new tables to LIB - not LIBNAME - because LIBNAME is the original library, whereas LIB is
the new library for output;
    put "data &lib..&memname;"
    ;
end;
&uqp._keeps = strip( strip( &uqp._keeps ) || ' ' || &uqp._name );
file &uqp._1;
put '    attrib    ' &uqp._name 'length = ' @;
if &uqp._type = 'C' then put '$' @;
put &uqp._length @;
if &uqp._format ne '@' then put 'format = ' &uqp._format @;
if &uqp._informat ne '@' then put 'informat = ' &uqp._informat @;
put ' ';
file &uqp._2;
if not first.&uqp._memname then put "    else if " @;
else put "    if " @;
put "&uqp._name = '" &uqp._name "' then " @;
if &uqp._type = "N" then do;
    &uqp._nums = strip( &uqp._nums ) || ' ' || &uqp._name;
    put &uqp._name "= &uqp._nvalue;";
end;
else do;
    &uqp._chars = strip( &uqp._chars ) || ' ' || &uqp._name;
    put &uqp._name "= &uqp._value;";
end;
if last.&uqp._memname then do;
    file &uqp._1;
    if &uqp._keeps = ' ' then link empty;
    else link nonempty;
end;
return;

empty:
put
    '    stop;'
    /'run;';
;
return;

nonempty:
put
    '    ;'
    /"    set work.&uqp._tmp;"
    /"    by %prefix(libname memname rcdseq varnum name type length varlen format informat,&uqp._);"
    /"    where &uqp._libname = '&libname' and upcase( &uqp._memname ) = '&memname';"
    /"    retain ";
do i = 1 to length( &uqp._keeps ) while ( scan( &uqp._keeps, i, ' ' ) ne ' ' );
    tmp = scan( &uqp._keeps, i, ' ' );
    put '    ' tmp;
end;
put
    '    ;'
    /"    if first.&uqp._rcdseq then do;"
    ;
if &uqp._nums ne ' ' then do;
    put
        "    array &uqp._nums ";
    do i = 1 to length( &uqp._nums ) while ( scan( &uqp._nums, i, ' ' ) ne ' ' );
        tmp = scan( &uqp._nums, i, ' ' );
        put '    ' tmp;
    end;
    put
        '    ;'
        /"    do over &uqp._nums; &uqp._nums = .; end;"
    ;
end;
if &uqp._chars ne ' ' then do;
    put
        "    array &uqp._chars ";
    do i = 1 to length( &uqp._chars ) while ( scan( &uqp._chars, i, ' ' ) ne ' ' );
        tmp = scan( &uqp._chars, i, ' ' );
        put '    ' tmp;
    end;
    put
        '    ;'
        /"    do over &uqp._chars; &uqp._chars = ' '; end;"
    ;
end;
put
    "    end;"
    ;
file &uqp._3;
put /"    if last.&uqp._rcdseq then output;"

```

```

        /" keep ";
        do i = 1 to length( &uqp._keeps ) while ( scan( &uqp._keeps, i, ' ' ) ne ' ' );
            tmp = scan( &uqp._keeps, i, ' ' );
            put ' ' tmp;
        end;
        put ' ' ;'
        /"run;"
    ;
    return;
run;

%filedump( &uqp._1 &uqp._2 &uqp._3 )

%include &uqp._1;
%include &uqp._2;
%include &uqp._3;
%end;

*In case there were any empty tables or catalogs (no records, no variables, or both), we need to use the
decoded METADATA to reconstruct them;
data _null_;
    file &uqp._0;
    set &lib..&metadata;
    where nob = 0 or nvar = 0 and upcase( memname ) ne "%upcase(&metadata)";
    by libname memname varnum;
    if first.memname then do;
        if nvar = 0 then do;
            put "data &lib.." memname ";";
            /" do i = 1 to " nob "; output; end;"
            /" drop i;"
            ;
        end;
        else if nob = 0 then do;
            put "data &lib.." memname ";";
            /" attrib;"
        end;
    end;
    if name ne ' ' then do;
        put name 'length = ' @;
        if type = 'C' then put '$' @;
        put length @;
        if format notin ( '@' ' ' ) then put 'format = ' format @;
        if informat notin ( '@' ' ' ) then put 'informat = ' informat @;
        put ';;';
    end;
    if last.memname then put
        ' stop;'
        /"run;"
        ;
run;

%include &uqp._0;

*Now make the format catalogs from the CNTLOUT tables;
data _null_;
    file &uqp._0;
    set &lib..&metadata;
    where memtype = 'CATALOG';
    by libname memname;
    if first.memname then put
        "data &templib1.." memname "; set &lib.." memname "; run;"
        /"proc delete data = &lib.." memname "; run;"
        /"proc format maxlablen = 200 library = &lib.." memname " cntlin = &templib1.." memname "; run;"
        ;
run;

%include &uqp._0;

*Now for the MEMLABELs and variable LABELs;
%prtempty( &lib..&metadata )
data _null_;
    file &uqp._1;
    set &lib..&metadata;
    by libname memname varnum;
    where upcase( memname ) ne "%upcase(&metadata)";
    if first.libname then put "proc datasets nolist library = &lib;";
    if first.memname then do;
        tmp = quote( trim( memlabel ) );
        put 'modify ' memname '( label = ' tmp ') / mt = ' memtype ';';
        if memtype ne 'CATALOG' and nvar > 0 then put ' label ';
    end;
    if memtype ne 'CATALOG' and nvar > 0 then do;
        tmp = quote( trim( label ) );
        put ' ' name '=' tmp;
        if last.memname then put ';;';
    end;
end;

```

```

        if last.libname then put 'quit;';
run;

%*Delete the METADATA table;
proc delete data = &lib..&metadata; run;

%filedump( &uqp._1 )
%include &uqp._1;

%goto fin;

%err:
    %put &err: Aborting...;
    %put &err: _local_;
    %unquote(e%str(ndsas;))

%fin:
    %put &id: End;
%mend denormal;

%*====;
%* START HERE
%*====;

    %if %index(.IN.OUT.,.&inout..) = 0 %then %do;
        %put &err: INOUT="&inout" not in ( IN OUT );
        %let kerr = 1;
    %end;
    %else %if %length(&uqp._1) > 8 or %sysfunc(nvalid(&uqp._1,V7)) ne 1 %then %do;
        %put &err: PREFIX=&prefix too long or invalid for use as fileref: &uqp._1;
        %let kerr = 1;
    %end;
    %else %if &lib = %then %do;
        %put &err: No libref specified;
        %let kerr = 1;
    %end;
    %else %if %sysfunc(libref(&lib)) ne 0 %then %do;
        %put &err: Libref LIB=&lib not assigned - SYMSG=&sysfunc(sysmsg());
        %let kerr = 1;
    %end;
    %else %do;
%*Ensure that PREFIX makes temporary table and variable names unique;
        %local ktab_collisions kcol_collisions kfmt_collisions;
        %let ktab_collisions = 0;
        %let kcol_collisions = 0;
        %let kfmt_collisions = 0;
        proc sql noprint;
            select count(*) as ktab_collisions
            into :ktab_collisions
            from dictionary.tables
            where libname = "%unquote(&lib)"
                  and upcase( memname ) like %unquote(%str('%')&prefix_escaped%str(^_%')) escape "^";

            select count(*) as kcol_collisions
            into :kcol_collisions
            from dictionary.columns
            where libname = "%unquote(&lib)"
                  and upcase( name ) like %unquote(%str('%')&prefix_escaped%str(^_%')) escape "^";

            select count(*) as kfmt_collisions
            into :kfmt_collisions
            from dictionary.catalogs
            where libname = "%unquote(&lib)" and upcase( objtype ) in ( 'FORMAT' 'FORMATC' 'INFMT' 'INFMTC' )
                  and upcase( objname ) like %unquote(%str('%')&prefix_escaped%str(^_%')) escape "^";

        quit;
        %if %eval(&ktab_collisions+&kcol_collisions+&kfmt_collisions) ne 0 %then %do;
            %put &err: PREFIX=&Prefix may collide with existing memnames/variables/format-entries in library
LIB=&lib;
            %let kerr = 1;
        %end;
    %end;
    %if &kerr ne 0 %then %goto err;

    %let &uqp._L1 = %sysfunc(pathname(work))\&uqp._L1;
    %let &uqp._L2 = %sysfunc(pathname(work))\&uqp._L2;
    %let &uqp._L3 = %sysfunc(pathname(work))\&uqp._L3;
    x "mkdir "&&&uqp._L1"";
    x "mkdir "&&&uqp._L2"";
    x "mkdir "&&&uqp._L3"";
    libname &uqp._L1 "&&&uqp._L1";
    libname &uqp._L2 "&&&uqp._L2";
    libname &uqp._L3 "&&&uqp._L3";

filename &uqp._0 "%sysfunc(pathname(work))\&uqp._tmp0.tmp";

```



```

proc datasets nolist kill library = work3; quit;

%xmlport( work3, inout = in )

%libcomp2( work1, work3, ignorevars =, ignorecase = N, excmems = cats dogs infmts )
%libcomp2( work1, work2, ignorevars =, ignorecase = N, excmems = cats dogs infmts )

%macro compfmtcats( lib1, lib2 );
  proc sql;
    create table tmp_catalogs as
    select distinct libname, memname
    from dictionary.catalogs
    where libname in ( "%upcase(&lib1)" "%upcase(&lib2)" )
      and objtype in ( "FORMAT" "FORMATC" "INFMT" "INFMTC" )
    order by memname, libname;
  quit;

  %prtempty( sashelp.vcatalog,
    where = libname in ( "%upcase(&lib1)" "%upcase(&lib2)" ) and memtype = "CATALOG" )

  data _null_;
    file tmp;
    set tmp_catalogs;
    by memname libname;
    if first.memname then put
      'proc format maxlablen = 200 library = &lib1..' memname 'cntlout = base; run;'
      '/proc sort data = base; by fmtname type hlo start end; run;'
      '/proc format maxlablen = 200 library = &lib2..' memname 'cntlout = compare; run;'
      '/proc sort data = compare; by fmtname type hlo start end; run;'
      '/proc compare outnoequal outbase outcomp base = base compare = compare out = diff fmt noprint; run;'
      '/%prtempty( diff_fmt, titles ="BASE=&lib1.." memname ', COMPARE=&lib2..' memname " " )'
      ;
  run;

  %include tmp;
%mend compfmtcats;

%compfmtcats( work1, work3 )
%compfmtcats( work1, work2 )

```